

Name: _____

Score: _____/40 _____%

Bits & Bytes: Level II – QUIZ

* Important Information:

- 1 byte = 8 bits.
- 1 kilobyte (KB) = 1,024 bytes.
- 1 megabyte (MB) = 1,024 kilobytes.
- 1 gigabyte (GB) = 1,024 megabytes.
- 1 terabyte (TB) = 1,024 gigabytes.
- All numbers are *unsigned* (non-negative) integers.

Multiple Choice Questions [1 point each]

- 1) To represent a number in *hexadecimal* means to represent it in base _____.
A) 6
B) 10
C) 16
D) 60
- 2) 2^N bytes = _____ bits.
A) 16^N
B) 2^{N-3}
C) 2^N
D) 2^{N+3}
- 3) _____ What is the highest unsigned integer you can store in a single byte?
A) 255
B) 100
C) 99
D) 9

* Hint: Convert the number, 1111 1111, from *binary* to *decimal*!

Codes	48	65	6C	6C	6F	20	57	6F	72	6C	64
Text											

15) [3½ points, ½ point for each correct code] Translate the following message from ASCII text into decimal ASCII character codes:

Text	J	e								
Codes			32							
Text	m	'	a	p	p	e	l	l	e	
Codes										32
Text	J	e	a	n	.					
Codes										

[How Many Bits/Bytes Do You Need ...](#)

16) [2 points] You are designing a system to encode characters as binary data. For example, a character, such as an uppercase A, is represented by a certain string of 0's and 1's, such as 00001. If your system needs to be able to encode uppercase letters (A–Z), lowercase letters (a–z), and digits (0–9), how many bits do you need (AT A MINIMUM) to store one character? No two characters can be represented by the same string of 0's and 1's! Every character must be represented by a string of the same *length* (number of bits)!

Don't worry about whether the number of bits is a multiple of 8!

_____ bits

* Hint: DON'T try to come up with all of the character codes!!! JUST count how many characters your system needs to be able to encode, and derive your answer from that number!

17) [2 points] You are programming a game, and need to use a *variable* (named location in memory) to store the player's score. If the player's score is guaranteed to be an integer between 0 and 10,000 points (inclusive), and your variable MUST take up a whole number of 8-bit bytes, how many bits of memory do you need to store the player's score?

_____ bits

18) [2 points] A *pointer* is a type of variable that can store a *memory address*. Every byte of memory, in a computer, has an *address*, which is a whole number that describes "where it is". The byte with an address of B is found B bytes "after" the "first" byte of memory, which has an address of 0.

- a. [1 point] How many different bytes of memory can you address with a 16-bit pointer?

_____ kilo / mega / giga / tera byte(s). Fill in the blank with a whole number LESS THAN 1,024, AND circle or highlight ONE of the prefixes.

- b. [1 point] How many different bytes of memory can you address with a 32-bit pointer?

_____ kilo / mega / giga / tera byte(s). Fill in the blank with a whole number LESS THAN 1,024, AND circle or highlight ONE of the prefixes.

[Paint by Numbers!](#)

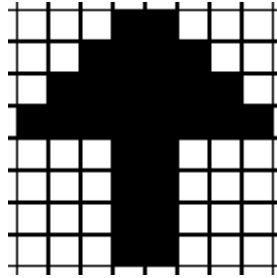
The following two questions delve into the topic of how *monochrome* (black-and-white) images are stored on a computer. The *MOI file format*, discussed below, is not a "real" image file format. However, it is useful in understanding "real" file formats that are used to store *uncompressed* digital images, such as the BMP file format.

- An MOI file starts with a *header*, which contains the following data about the image:
 - A 1-byte number: the image's width (in *pixels*). Call this number "W".
 - Another 1-byte number: the image's height (in *pixels*). Call this number "H".
- The header is followed by the pixels that make up the "W × H image":
 - An image consists of H *rows* of pixels.
 - Each row of the image consists of W pixels.
 - Each pixel takes up 1 bit of space:
 - A black pixel gets stored as a 0 bit.
 - A white pixel gets stored as a 1 bit.
 - NO extra bits get inserted between pixels.
 - NO extra bits get inserted between rows of pixels.

- The image's pixels will require at least $W \times H$ bits.
 - If $W \times H$ is evenly divisible by 8 (the number of bits in a byte), you don't need to add any extra bits after the image's pixels.
 - **HOWEVER, if $W \times H$ is NOT evenly divisible by 8, you need to *append* 0 bits to the end of the last byte of the file, until that byte has 8 bits in it.**

In the following questions, you will represent the bytes of each image in *hex*. Instead of using 8 binary digits (bits!) to represent a byte, you will use 2 hexadecimal digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F). For example, you would write the *nibble* (string of 4 bits, or half-byte), 0110, as 6. You would write the byte, 1110 0111, as E7. Hex is just a more compact way of writing binary data!

19) [2½ points, ¼ point for each correct byte] Represent the following 8×8 MOI image in hex:



20) [6½ points, ¼ point for each correct byte] Represent the following 12 × 16 MOI image in hex:

